Hadoop and HBase on the Cloud:

A Case Study on Performance and Isolation.

Konstantin V. Shvachko ***Jagane Sundar***

June 14, 2012

WANdisco

ApacheCon North America
Portland, Oregon 26th – 28th February 2013

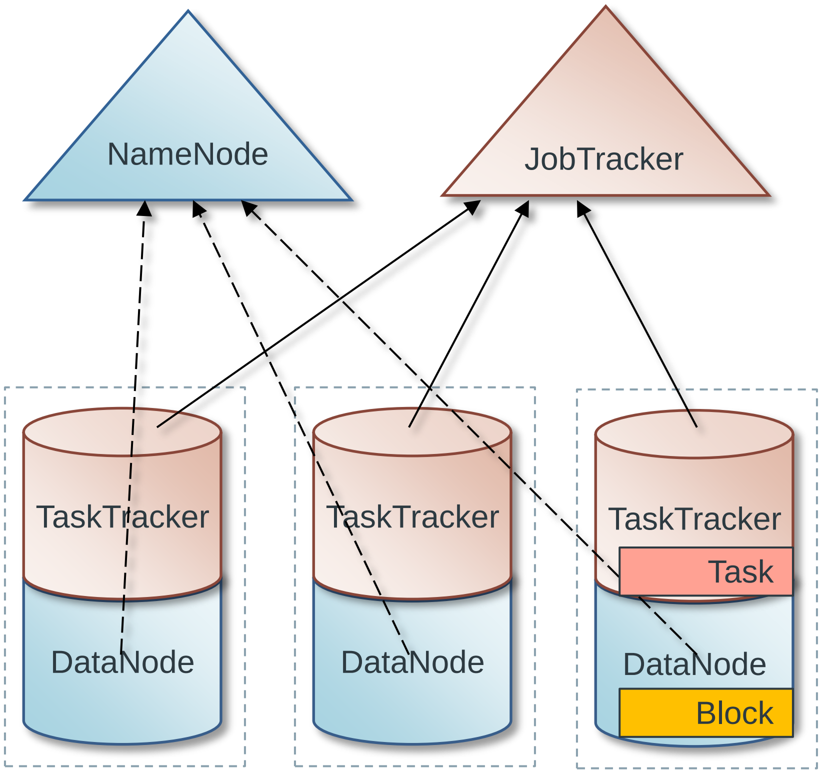
Authors

- Founders of AltoStor and AltoScale
- Jagane: WANdisco, CTO and VP Engineering of Big Data
  - Director of Hadoop Performance and Operability at Yahoo!
  - Big data, cloud, virtualization, and networking experience
- Konstantin: WANdisco, Chief Architect
  - Hadoop, HDFS at Yahoo! & eBay
  - Efficient data structures and algorithms for large-scale distributed storage systems
  - Giraffa - file system with distributed metadata & data utilizing HDFS and HBase. Hosted on Apache Extra

What is Apache Hadoop

A reliable, scalable, high performance distributed computing system

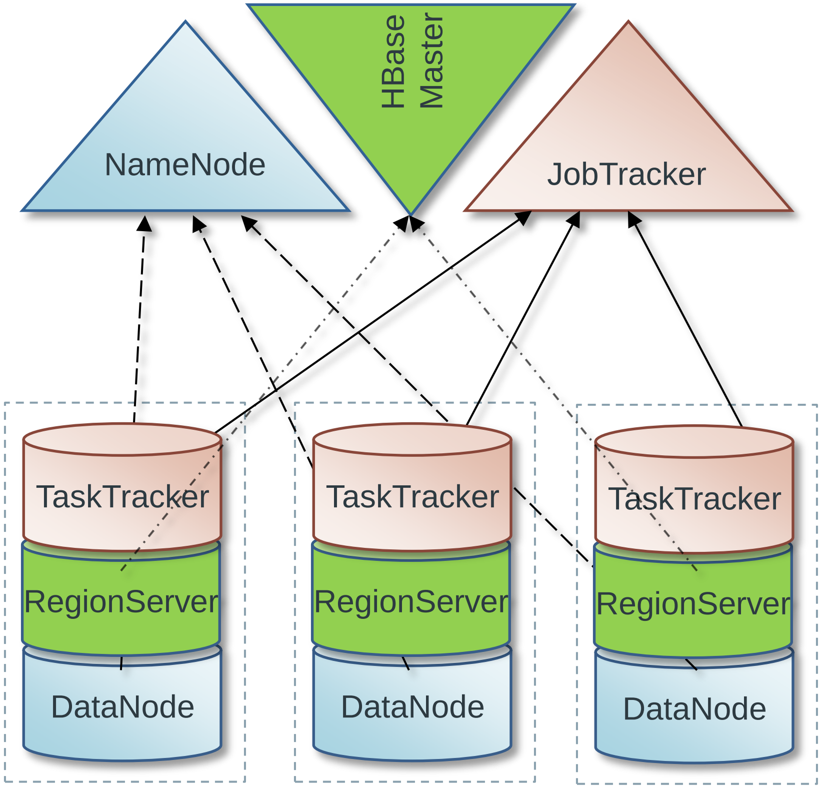
- The Hadoop Distributed File System (HDFS)
  - Reliable storage layer
  - NameNode – namespace and block management
  - DataNodes – block replica container
- MapReduce – distributed computation framework
  - Simple computational model
  - JobTracker – job scheduling, resource management, lifecycle coordination
  - TaskTracker – task execution module
- Analysis and transformation of very large amounts of data using commodity servers



What is Apache HBase

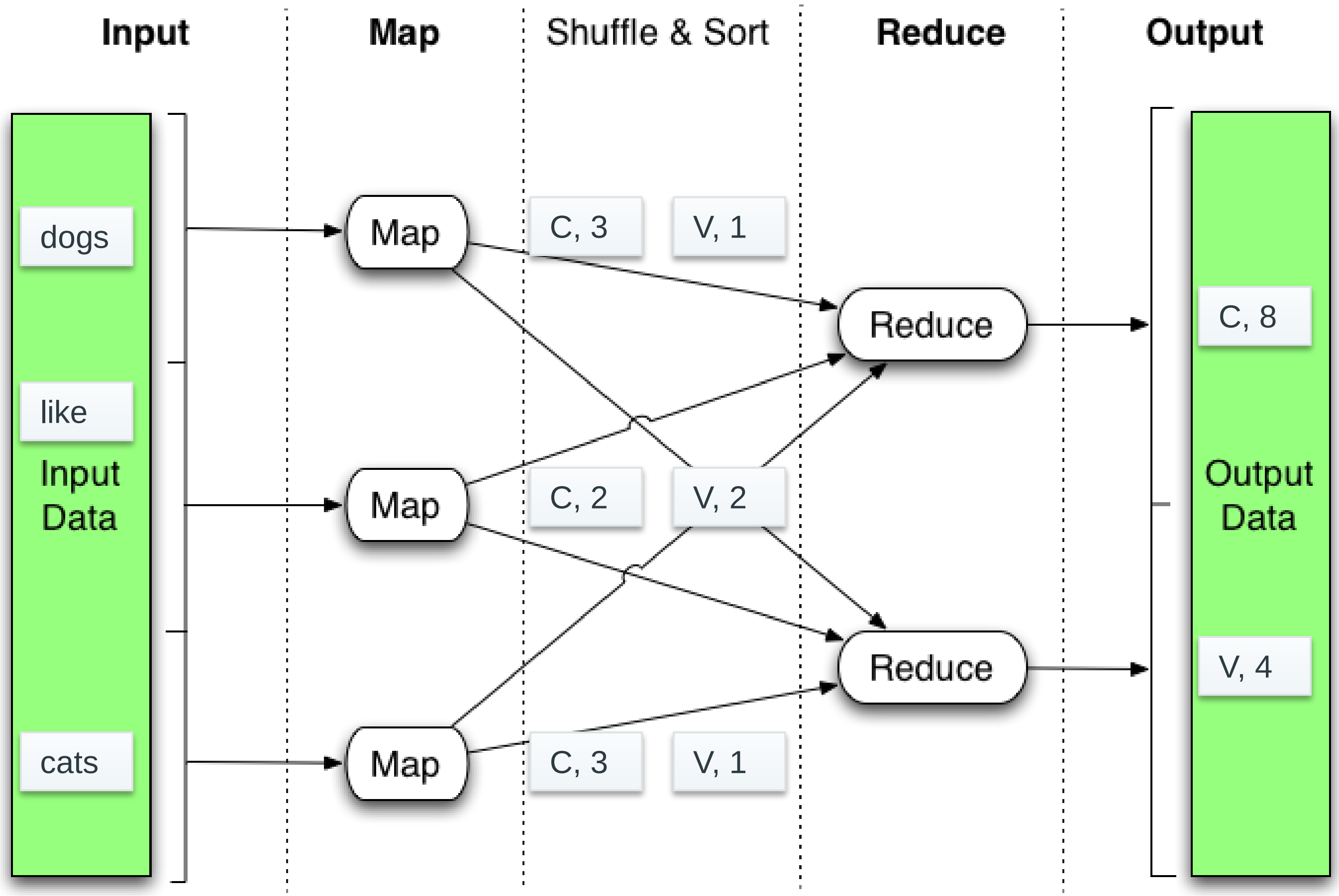
A distributed key-value *storage for real-time access to semi-structured data*

- Table: big, sparse, loosely structured
  - Collection of rows, sorted by *row keys*
  - Rows can have arbitrary number of columns
- Table is split Horizontally into Regions
  - Dynamic Table partitioning
  - Region Servers serve regions to applications
- Columns grouped into Column families
  - Vertical partition of tables
- ***Distributed Cache:*** Regions are loaded in nodes' RAM
  - Real-time access to data



What is MapReduce

A Parallel Computational Model and Distributed Framework



What is the Problem

Low Average CPU Utilization on Hadoop Clusters

- I/O utilization
  - Can run with the speed of spinning drives
  - Examples: DFSIO, Terasort (well tuned)
- Network utilization – optimized by design
  - Data locality. Tasks executed on nodes where input data resides. No massive transfers
  - Block replication of 3 requires two data transfers
  - Map writes transient output locally
  - Shuffle requires cross-node transfers
- CPU utilization
  1. IO bound workloads preclude from using more cpu time
  2. Cluster provisioning:
     peak-load performance vs. average utilization tradeoff

CPU Load

- Computation of Pi
  - pure CPU workload, no input or output data
  - Enormous amount of FFTs computing amazingly large numbers
  - Record Pi run over-heated the datacenter
- Well tuned Terasort is CPU intensive
- Compression – marginal utilization gain
- Production clusters run cold
  1. IO bound workloads
  2. Conservative provisioning of cluster resources to meet strict SLAs

Two quadrillionth (10^{15}) digit of π is 0

Cluster Provisioning Dilemma

Rule of thumb

- 72 GB - total RAM / node
  - 4 GB – DataNode
  - 2 GB – TaskTracker
  - 16 GB – RegionServer
  - 2 GB – per individual task: 25 task slots (17 maps and 8 reduces)
- Average utilization vs peak-load performance
  - Oversubscription (28 task slots)
  - Better average utilization
  - MR Tasks can starve HBase RegionServers
- Better Isolation of resources ⟶ Aggressive resource allocation

Increasing IO Rate

With non-spinning storage

- Goal: Eliminate disk IO contention
- Faster non-volatile storage devices improve IO performance
  - Advantage in random reads
  - Similar performance for sequential IOs
- More RAM: HBase caching

What is DFSIO

Standard Hadoop Benchmark measuring HDFS performance

- DFSIO benchmark measures average throughput for IO operations
  - Write
  - Read (sequential)
  - Append
  - Random Read (new)
- MapReduce job
  - Map: same operation write or read for all mappers. Measures throughput
  - Single reducer: aggregates the performance results
- Random Reads (MAPREDUCE-4651)
  - *Random Read DFSIO* randomly chooses an offset
  - *Backward Read DFSIO* reads files in reverse order
  - *Skip Read DFSIO* reads seeks ahead after every portion read
  - Avoid read-ahead buffering
  - Similar results for all three random read modifications

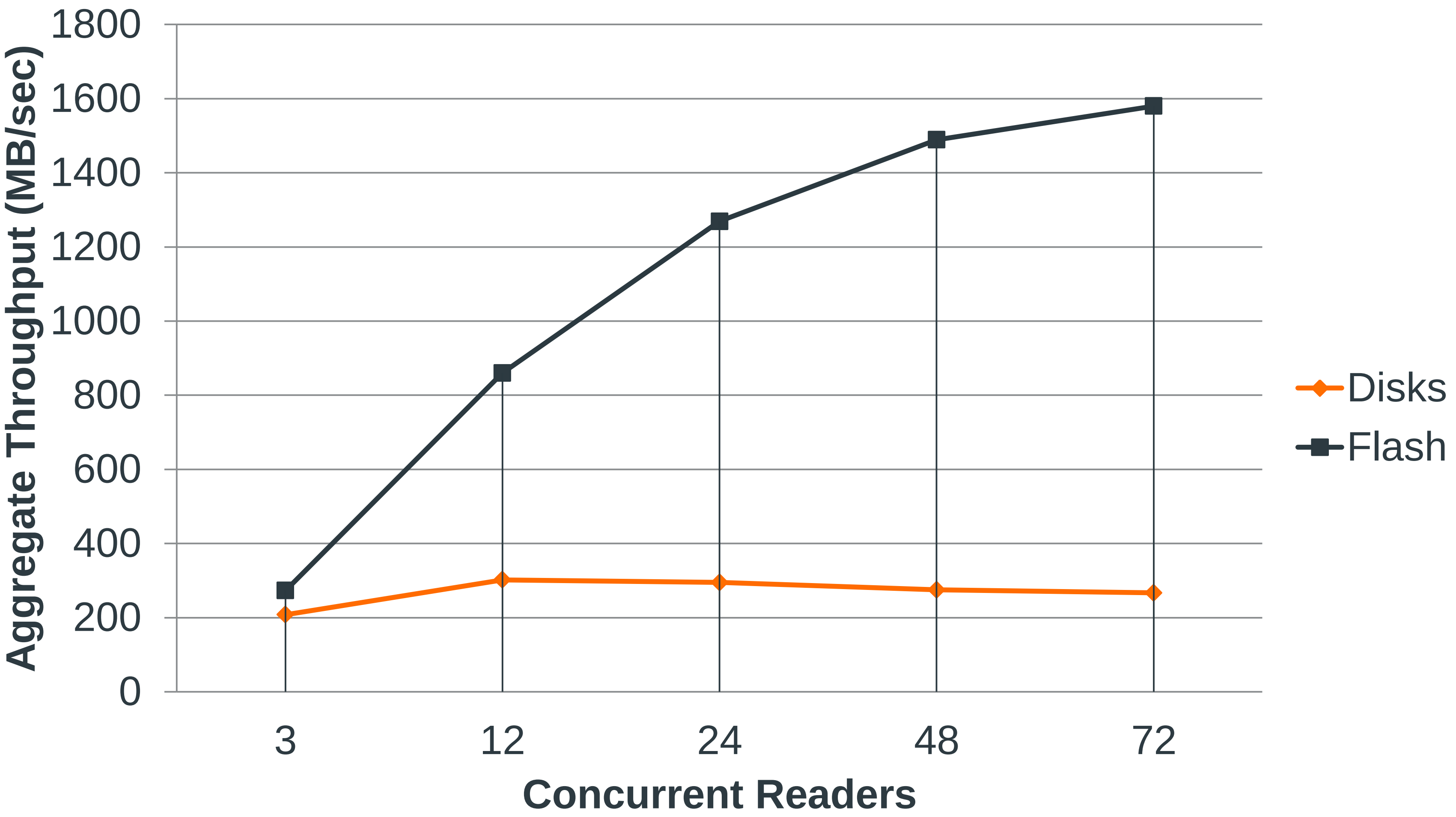
Benchmarking Environment

DFSIO

- Four node cluster: *Hadoop 1.0.3* *HBase 0.92.1*
  - 1 master-node: NameNode, JobTracker
  - 3 slave node: DataNode, TaskTracker
- Node configuraiton
  - Intel 8 core processor with hyper-threading
  - 24 GB RAM
  - Four 1TB 7200 rpm SATA drives
  - 1 Gbps network interfaces
- DFSIO dataset
  - 72 files of size 10 GB each
  - Total data read: 7GB
  - Single read size: 1 MB
  - Concurrent readers: from 3 to 72

Random Reads

Increasing Load with Random Reads



What is YCSB

Yahoo! Cloud Serving Benchmark

- YCSB allows to define a mix of read / write operations, measure latency and throughput
  - Compares different database: relational and no-SQL
  - Data is represented as a table of records with number of fixed fields
  - Unique key identifies each record
- Main operations
  - *Insert*: Insert a new record
  - *Read*: Read a record
  - *Update*: Update a record by replacing the value of one field
  - *Scan*: Scan a random number of consequent records, starting at a random record key

Benchmarking Environment

YCSB

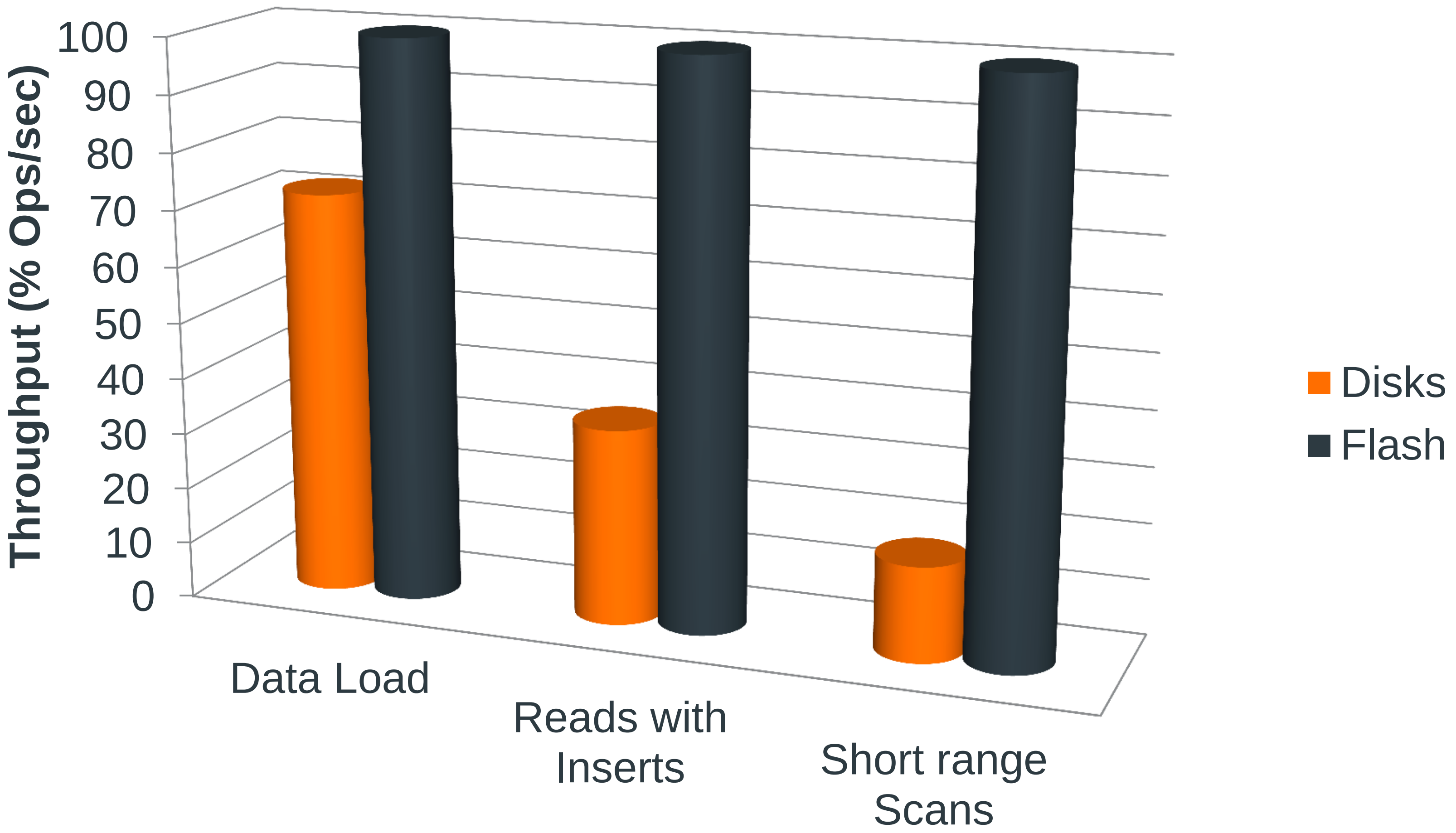
- Four node cluster
  - 1 master-node: NameNode, JobTracker, HBase Master, Zookeeper
  - 3 slave node: DataNode, TaskTracker, RegionServer
  - Physical master node
  - 2 to 4 VMs on a slave node. Max 12 VMs
- YCSB datasets of two different sizes: 10 and 30 million records
  - dstat collects system resource metrics: CPU, memory usage, disk and network stats

YCSB Workloads

Workloads	Insert %	Read %	Update %	Scan %
Data Load	100			
Reads with heavy insert load	55	45		
Short range scans: workload E	5			95

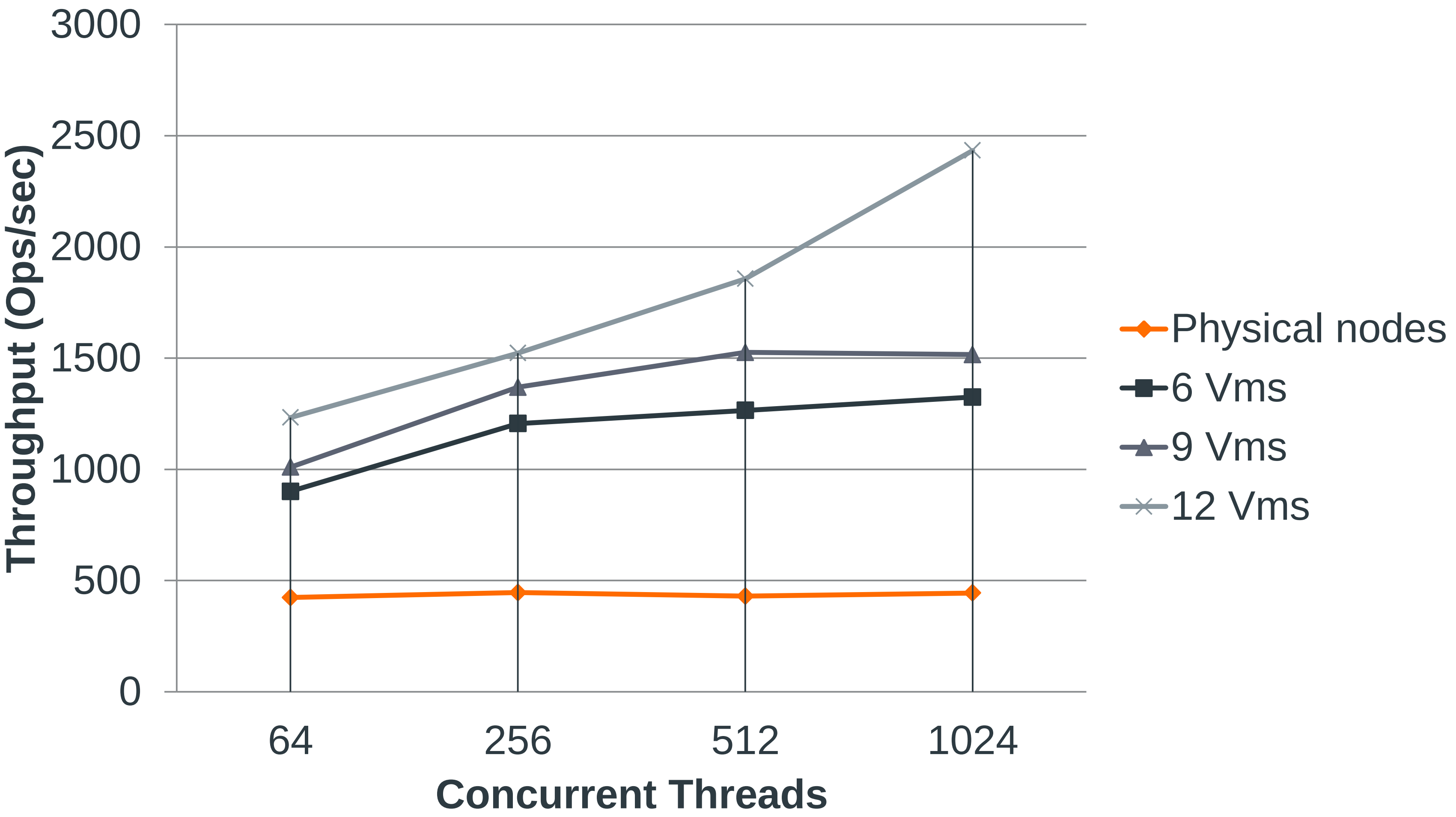
Average Workloads Throughput

Random reads and Scans substantially faster with flash



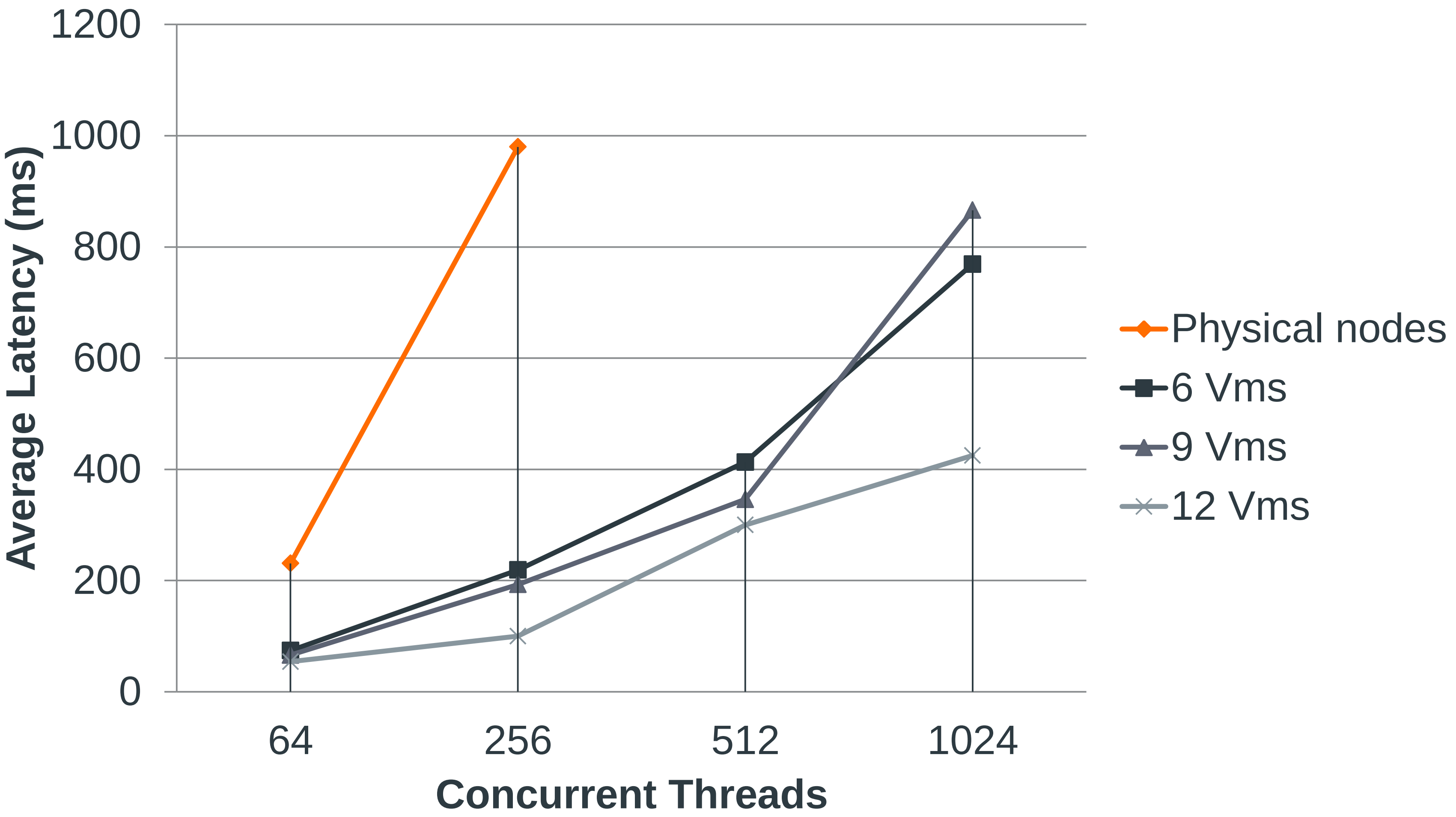
Short range Scans: Throughput

Adding one VM per node increases overall performance 20% on average



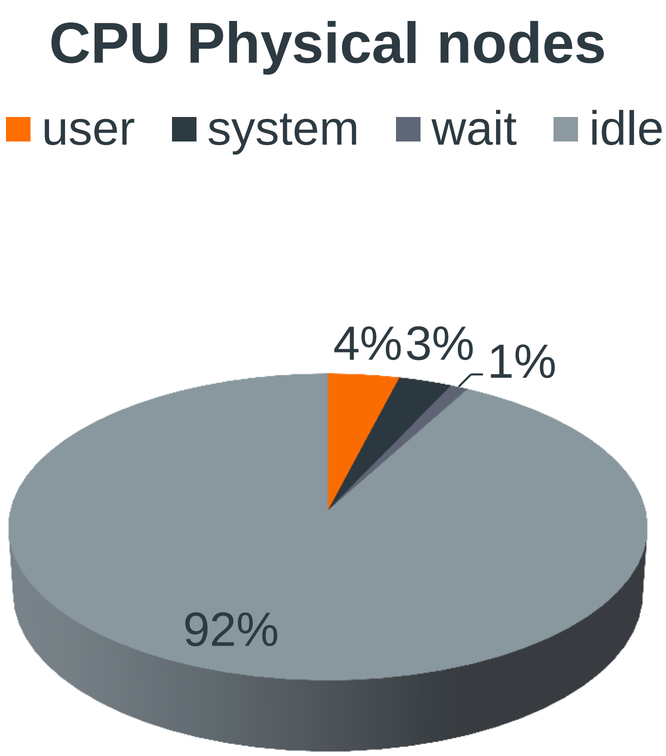
Short range Scans: Latency

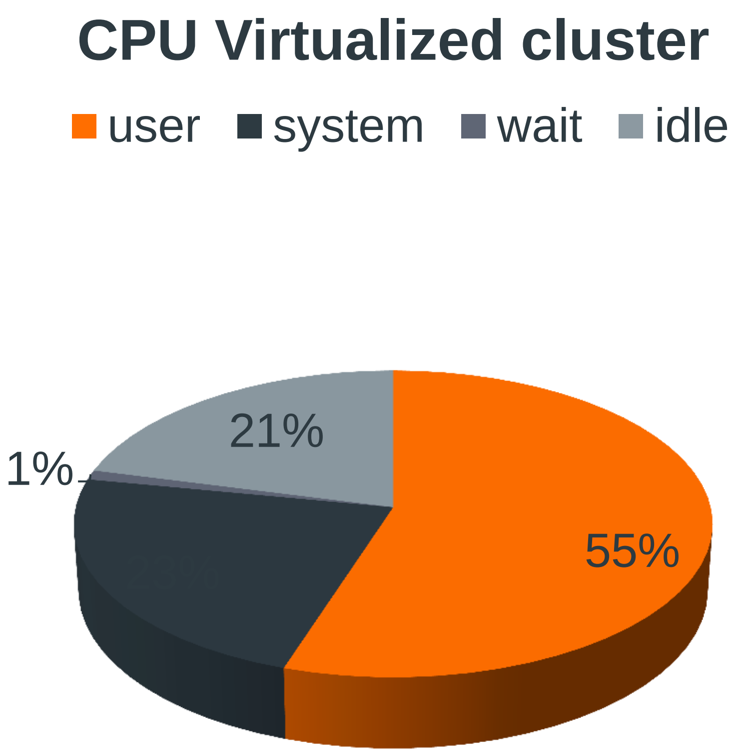
Latency grows linearly with number of threads on physical nodes



CPU Utilization comparison

Virtualized Cluster drastically increases CPU utilization

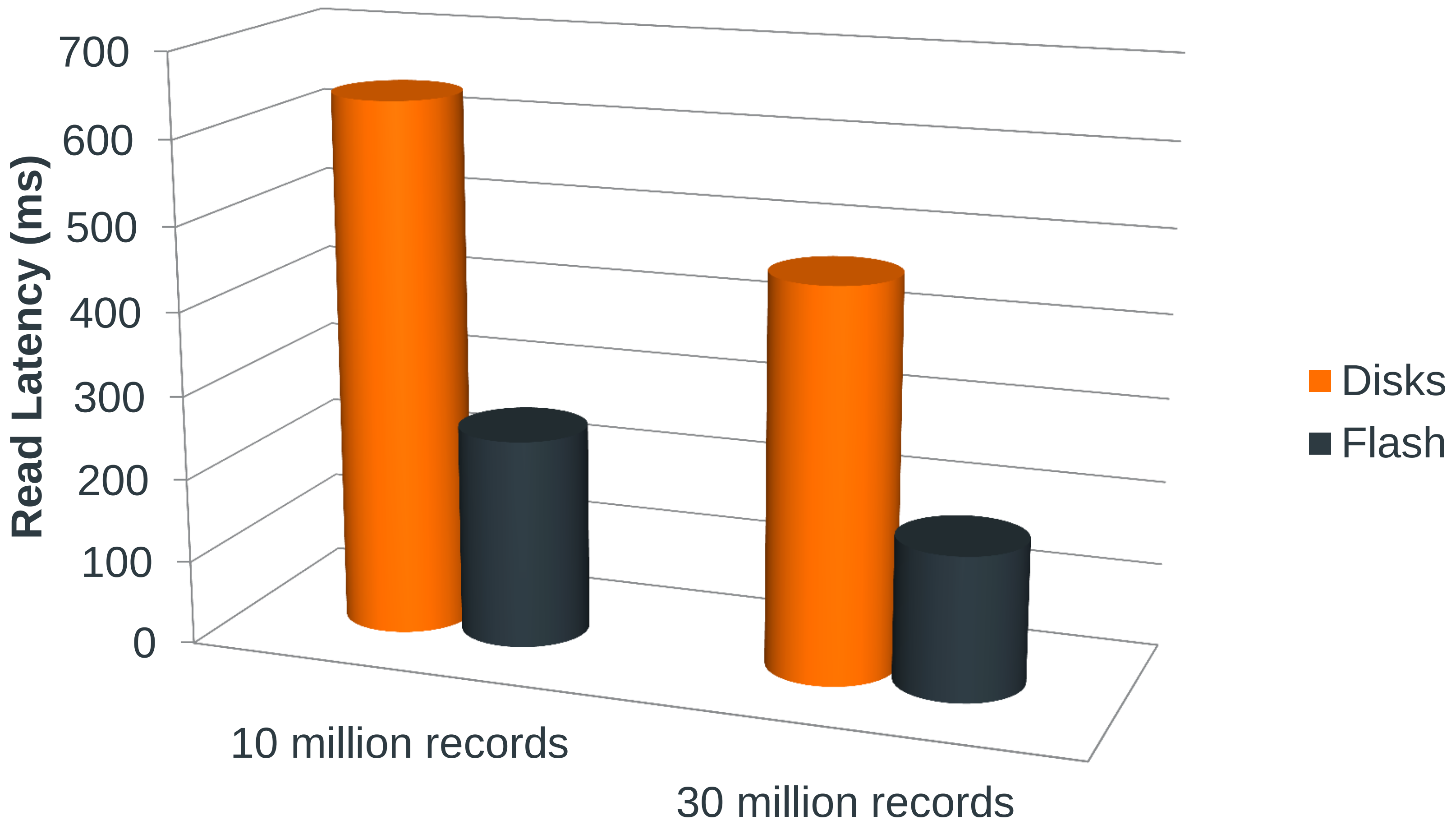




- Physical node cluster generates very light CPU load – 92% idle
- With VMs the CPU can be drawn close to 100% at peaks

Reads with Inserts

Latency of reads on mixed workload: 45% reads and 55% inserts



Conclusions

VMs allow to utilize Random Read advantage of flash for Hadoop

- HDFS
  - *Sequential IO* is handled well by the disk storage
  - Flash substantially outperforms disks on workloads with *random reads*
- HBase *write-only workload* provides marginal improvement for flash
- Using *multiple VMs* / node provides *100% peak utilization* of HW resources
  - CPU utilization on physical-node clusters is a fraction of its capacity
- Combination of Flash Storage and Virtualization implies high performance of HBase for *Random Read* and *Reads Mixed with writes* workloads
- **Virtualization** serves two main functions:
  - **Resource utilization** by running more server processes per node
  - **Resource isolation** by designating certain percentage of resources to each server and not letting them starve each other

Thank you

Konstantin V. Shvachko

Jagane Sundar